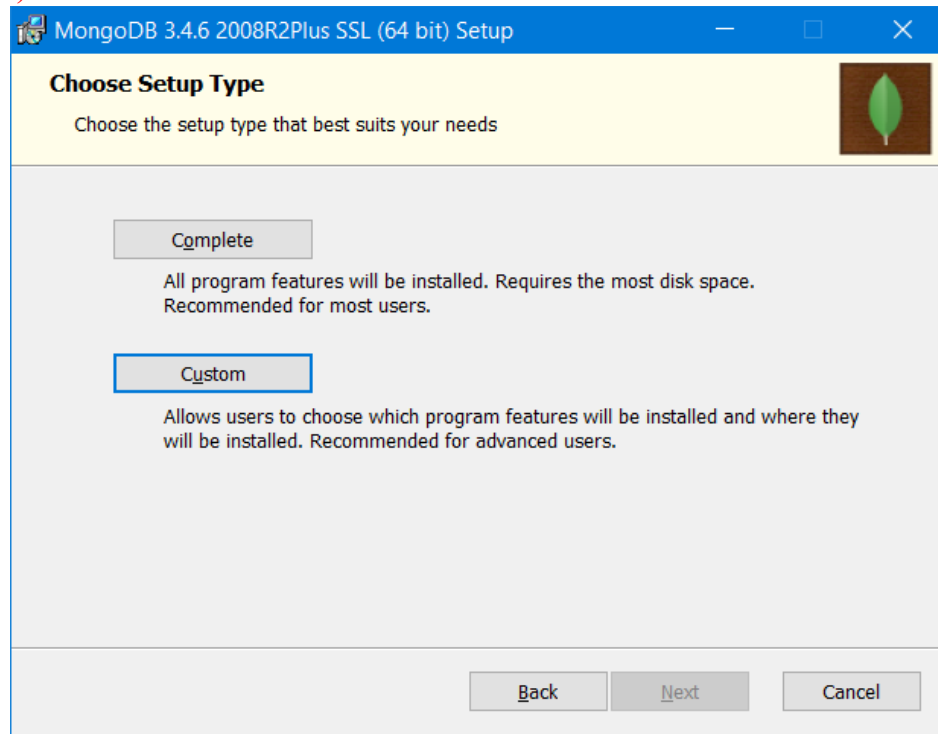

BÀI TẬP MONGODB

CÀI ĐẶT VÀ CẤU HÌNH MONGODB

CÀI ĐẶT

Download Mongo tại: <https://www.mongodb.com/download-center?jmp=nav>


Tiến hành cài đặt bình thường (chú ý: **NÊN** chọn thư mục cài đặt là **C:\mongodb** thay vì vào **Program Files**)



MongoDB 3.4.6 2008R2Plus SSL (64 bit) Setup

Change destination folder


Browse to the destination folder




Look in:

3.4

▼





Folder name:

C:\MongoDB\Server\3.4\

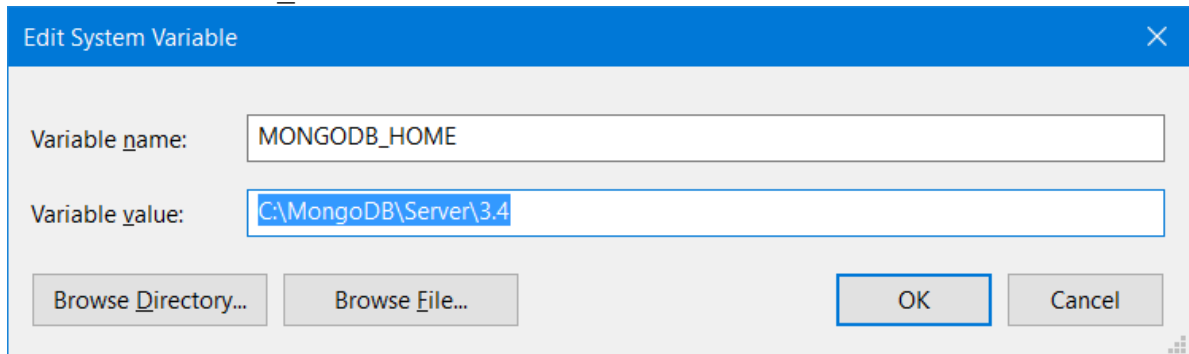
OK

Cancel

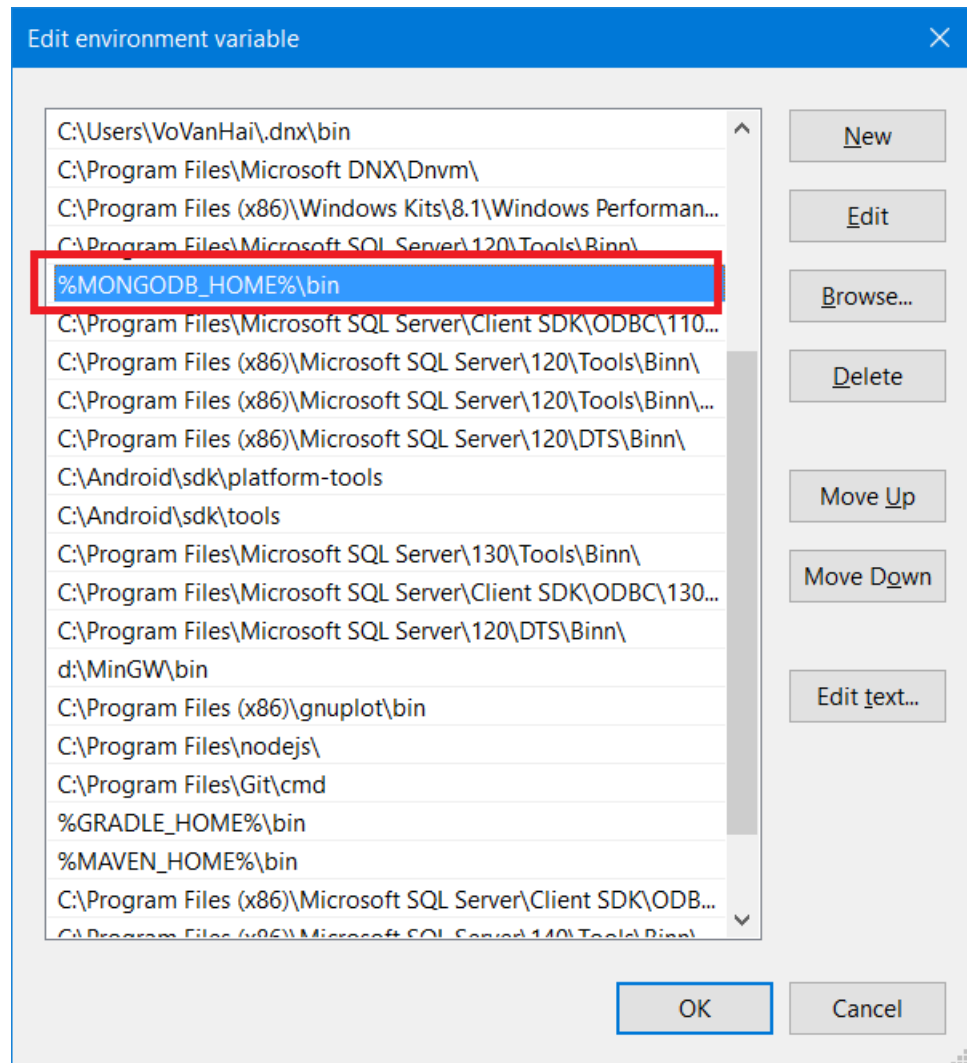
CẤU HÌNH

Thiết lập biến môi trường

Thêm biến MONGODB_HOME



Thay đổi biến PATH



KHOI ĐỘNG MONGODB SERVER

SỬ DỤNG COMMAND-LINE

Start mongo server với 1 cổng bất kỳ:

```
mongod --port 9999 --dbpath data --logpath logs\mglog.log --logappend
```

Cổng mặc định là: 27017

```
mongod --dbpath data --logpath logs\mglog.log --logappend
```

ĐĂNG KÝ MỘT WINDOWS SERVICE

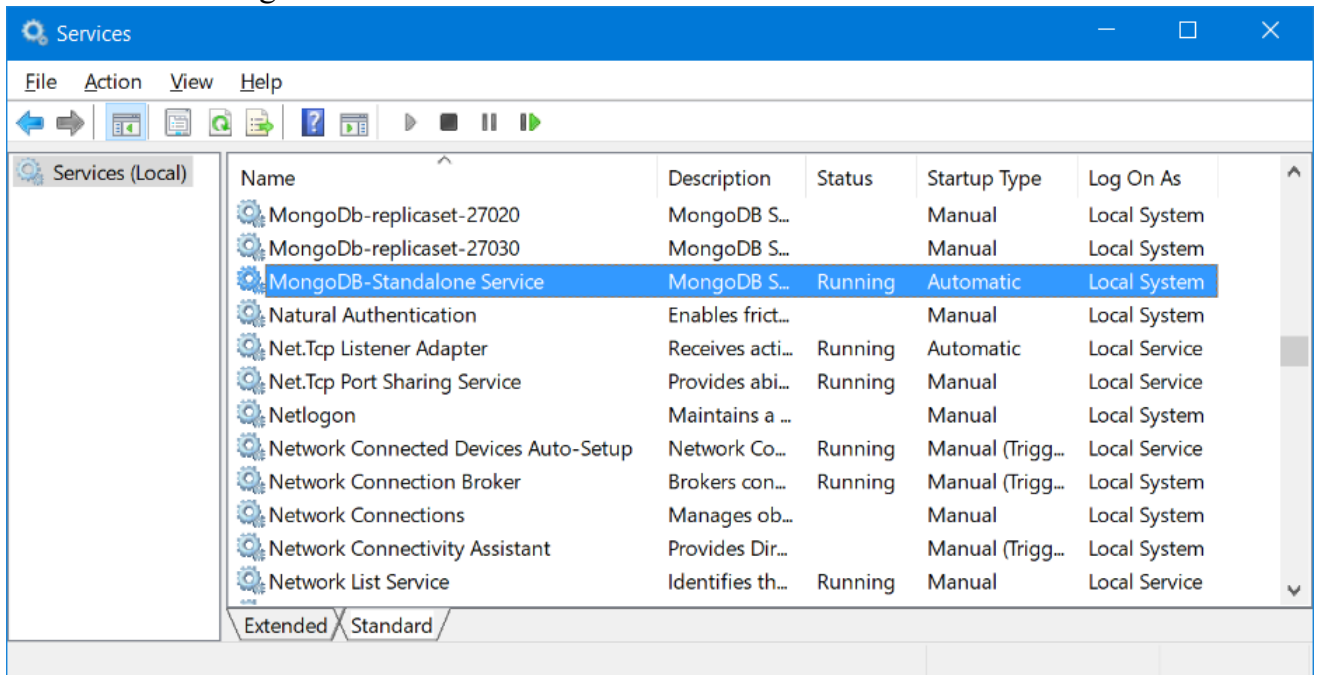
Tạo file config tên solo.cfg (đường dẫn trực tiếp D:\mongo-solo\solo.cfg)

```
dbpath=D:\mongo-solo\data
logpath=D:\mongo-solo\logs\solo.log
logappend=true
port = 27017
```

Tạo service với lệnh

```
mongod --config "D:\mongo-solo\solo.cfg" --install --serviceName "MongoDb" --
serviceDisplayName "MongoDB-Standalone Service"
```

Mở services manager



Nhấn phải chuột chọn start hoặc chạy command-line

```
net start MongoDB
```

Stop service chạy command-line

```
net stop MongoDB
```

Để xóa service đã tạo, chạy command-line

```
sc delete MongoDB
```

THAO TÁC CLIENT

KHỞI ĐỘNG CLIENT

mongo localhost:port

Hoặc đơn giản chạy công mặc định

mongo



```
Command Prompt - mongo
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\VoVanHai>mongo
MongoDB shell version v3.4.6
connecting to: mongod://127.0.0.1:27017
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-26T09:38:16.191+0700 I CONTROL [initandlisten]
2017-07-26T09:38:16.192+0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-26T09:38:16.192+0700 I CONTROL [initandlisten] **           Read and write access to data and configuration is unrestricted.
2017-07-26T09:38:16.192+0700 I CONTROL [initandlisten]
>
```

CÁC THAO TÁC

CƠ BẢN

Liệt kê tất cả các database

```
> show dbs
AdventureWorks  0.006GB
local           0.000GB
sampledb       0.002GB
>
```

Chuyển quyền sử dụng sang một database (không có thì tạo mới)

```
> use sampledb
switched to db sampledb
>
```

Hiển thị tất cả các collections có trong database đó

```
> show collections
zips
>
```

Hiển thị trợ giúp

```
Command Prompt - mongo

> help

db.help()                help on db methods
db.mycoll.help()          help on collection methods
sh.help()                 sharding helpers
rs.help()                 replica set helpers
help admin                administrative help
help connect              connecting to a db help
help keys                 key shortcuts
help misc                 misc things to know
help mr                   mapreduce

show dbs                  show database names
show collections           show collections in current database
show users                show users in current database
show profile              show most recent system.profile entries with time >= 1ms
show logs                 show the accessible logger names
show log [name]           prints out the last segment of log in memory, 'global' is default
use <db_name>             set current database
db.foo.find()             list objects in collection foo
db.foo.find( { a : 1 } )  list objects in foo where a == 1
it                        result of the last line evaluated; use to further iterate
DBQuery.shellBatchSize = x set default number of items to display on shell
exit                     quit the mongo shell

> _
```

CREATE OPERATIONS

Dùng các lệnh sau

`db.collection.insertOne()` để chèn 1 document

`db.collection.insertMany()` để chèn nhiều document

Ví dụ:

```
db.users.insertOne(      ← collection
{
  name: "sue",           ← field: value
  age: 26,                ← field: value
  status: "pending"      ← field: value
})                        } document
```

READ OPERATIONS

Sử dụng lệnh

`db.collection.find()`

để lấy hết tất cả các document. Tuy nhiên ta có thể giới hạn kết quả theo các tiêu chí

Ví dụ

```
db.users.find(           ← collection
  { age: { $gt: 18 } },   ← query criteria
  { name: 1, address: 1 } ← projection
).limit(5)               ← cursor modifier
```

UPDATE OPERATIONS

Sử dụng các lệnh sau

db.collection.updateOne()

db.collection.updateMany()

db.collection.replaceOne()

Ví dụ

```
db.users.updateMany(
  { age: { $lt: 18 } },
  { $set: { status: "reject" } }
)
```

← collection
← update filter
← update action

Delete Operations

Sử dụng các lệnh sau

db.collection.deleteOne()

db.collection.deleteMany()

Ví dụ

```
db.users.deleteMany(
  { status: "reject" }
)
```

← collection
← delete filter

IMPORT DATA

Import dữ liệu file json (zips.json) vào mongo

```
mongoimport --host localhost --port 27017 --db sampled -  
-collection zips --file zips.json
```

SQL SERVER VS. MONGODB

Một số khái niệm

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	\$lookup, embedded documents
primary key Specify any unique column or column combination as primary key.	primary key In MongoDB, the primary key is automatically set to the _id field.
aggregation (e.g. group by)	aggregation pipeline See the SQL to Aggregation Mapping Chart.

Đọc thêm tại: <https://docs.mongodb.com/manual/reference/sql-comparison/>

BÀI TẬP LẬP TRÌNH

Bài 1: Thực hiện lại các bước phân hướng dẫn

Bài 2: Cho file zips.json. Import vào database có tên zipsdb, collection có tên zips.

Sau đó thực hiện các yêu cầu:

- Hiển thị tất cả các documents
- Chèn thêm 1 document mới
- Tìm các documents có city là PALMER
- Tìm các documents có dân số >100000
- Tìm dân số của thành phố FISHERS ISLAND
- Tìm các thành phố có dân số từ 10 - 50
- Tìm tất cả các thành phố của bang MA có dân số trên 500
- Tìm tất cả các bang (distinct)
- Tìm tất cả các bang mà có chứa ít nhất 1 thành phố có dân số trên 100000
- Tính dân số trung bình của mỗi bang
- Bang WA có bao nhiêu city
- Tính số city của mỗi bang
- Tìm tất cả các bang có tổng dân số trên 10000000

THAO TÁC VỚI MONGODB

Cho tài liệu về các nhà hàng được lưu tại:

<http://www.w3resource.com/mongodb-exercises/retaurants.zip>.

Cấu trúc của tài liệu như sau:

```
1 {
2   "address": {
3     "building": "1007",
4     "coord": [ -73.856077, 40.848447 ],
5     "street": "Morris Park Ave",
6     "zipcode": "10462"
7   },
8   "borough": "Bronx",
9   "cuisine": "Bakery",
10  "grades": [
11    { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
12    { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
13    { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
14    { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
15    { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
16  ],
17  "name": "Morris Park Bake Shop",
18  "restaurant_id": "30075445"
19 }
```

Download tài liệu về, tiến hành import vào mongodb rồi thực hiện các yêu cầu sau:

1. Write a MongoDB query to display all the documents in the collection restaurants.
2. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.
3. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine, but exclude the field _id for all the documents in the collection restaurant.
4. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.
5. Write a MongoDB query to display all the restaurant which is in the borough Bronx.
6. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.
7. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.
8. Write a MongoDB query to find the restaurants who achieved a score more than 90.
9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.
10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.
11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.
12. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and not located in the longitude less than -65.754168.

Note : Do this query without using \$and operator.

13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.
15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.
16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.
17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.
18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.
19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronx or Brooklyn.
20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.
21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.
22. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates.
23. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".
24. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52.
25. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.
26. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
27. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.
28. Write a MongoDB query to know whether all the addresses contains the street or not.
29. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.
30. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.
31. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.
32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

Nguồn và lời giải có thể tìm thấy ở đây: <http://www.w3resource.com/mongodb-exercises/>

CODE

DRIVERS

Download: <https://mongodb.github.io/mongo-java-driver/>

Maven dependencies

```
<dependencies>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver</artifactId>
    <version>3.4.2</version>
  </dependency>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongo-java-driver</artifactId>
    <version>3.4.2</version>
  </dependency>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-async</artifactId>
    <version>3.4.2</version>
  </dependency>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-core</artifactId>
    <version>3.4.2</version>
  </dependency>
</dependencies>
```

Driver Core:

<https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver-core>

Driver Async

<https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver-async/>

BSON

<https://oss.sonatype.org/content/repositories/releases/org/mongodb/bson/3.4.2/>

KẾT NỐI MONGODB

KẾT NỐI MONGODB VỚI SYNC DRIVER

Kết nối server đơn

```
MongoClient mongoClient = new MongoClient("localhost" , 27017 );
MongoDatabase db = mongoClient.getDatabase("mondial");
MongoCollection<Document> col = db.getCollection("countries");
FindIterable<Document> iter = col.find();
iter.iterator().forEachRemaining(t->{
    if(t.get("Name").toString().equalsIgnoreCase("Vietnam"))
        System.out.println(t.toString());
});
mongoClient.close();
```

Kết nối replicaset

```
List<ServerAddress>svrs=new ArrayList<>();
svrs.add(new ServerAddress("localhost:27017"));
svrs.add(new ServerAddress("localhost:27020"));
svrs.add(new ServerAddress("localhost:27030"));
List<MongoCredential> cres=new ArrayList<>();
cres.add(MongoCredential.createCredential("admin", "mondial", "admin".toCharArray()));
MongoClient mongoClient = new MongoClient(svrs , cres );
```

KẾT NỐI MONGODB VỚI ASYNC DRIVER

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CountDownLatch;
import org.bson.Document;
import com.mongodb.Block;
import com.mongodb.ConnectionString;
import com.mongodb.ServerAddress;
import com.mongodb.async.SingleResultCallback;
import com.mongodb.async.client.FindIterable;
import com.mongodb.async.client.MongoClient;
import com.mongodb.async.client.MongoClientSettings;
import com.mongodb.async.client.MongoClients;
import com.mongodb.async.client.MongoCollection;
import com.mongodb.async.client.MongoDatabase;
import com.mongodb.connection.ClusterSettings;
import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Sorts.*;

public class ConnectStandaloneServer {

    public static void main(String[] args)throws Exception {
        MongoClient mongoClient=null;

        // To directly connect to the default server localhost on port 27017
        mongoClient = MongoClients.create();

        // Use a Connection String
        mongoClient = MongoClients.create("mongodb://localhost");

        // or a Connection String
        mongoClient = MongoClients.create(new
        ConnectionString("mongodb://localhost"));
```

```

// or provide custom MongoClientSettings
ClusterSettings clusterSettings = ClusterSettings.builder().hosts(
    Arrays.asList(new ServerAddress("localhost:27017"))
).build();
MongoClientSettings settings = MongoClientSettings.builder()
    .clusterSettings( clusterSettings).build();
mongoClient = MongoClient.create(settings);

final CountDownLatch latch = new CountDownLatch(1);

MongoDatabase db = mongoClient.getDatabase("mondial");
MongoCollection<Document> coll = db.getCollection("countries");

FindIterable<Document> iter = coll.find();

iter.forEach(new Block<Document>() {
    public void apply(Document doc) {
        System.out.println(doc);
    }
}, new SingleResultCallback<Void>() {
    @Override
    public void onResult(Void v, Throwable t) {
        System.out.println("finish");
        if (t != null) {
            System.out.println("listDatabaseNames() errored: " +
t.getMessage());
        }
        latch.countDown();//ngừng tiến trình đợi khi load xong
    }
});

latch.await();//đợi cho công việc hoàn tất
mongoClient.close();
}
}

```

THAO TÁC CRUD VỚI SYNC DRIVER

CREATE

```

//chèn document
void insert(MongoCollection<Document> col) {
    Document lop=new Document("malop", "DHTH10A")
        .append("tenlop", "Lớp DH Kỹ Thuật PHẦN MỀM 10A");
    col.insertOne(lop);
}

void insert2( MongoDB database) {
    //chèn bằng đối tượng BasicDBObject
    MongoCollection<BasicDBObject> collection =
        database.getCollection("lophoc", BasicDBObject.class);
    Map<String,String> map =new HashMap<>();
    map.put("malop", "CDTH9LT");
    map.put("tenlop", "Lớp cao đẳng 9 liên thông");
    BasicDBObject bo1=new BasicDBObject(map);
    collection.insertOne(bo1);
}

```

READ

```

void findBymalop(MongoCollection<Document> col,String malop){
    Document doc = col.find(eq("malop", malop)).first();
}

```

```

    System.out.println(doc);
}

```

UPDATE

```

void update(MongoCollection<Document> col) {
    //đổi tượng dữ liệu mới cần cập nhật. Lưu ý $set
    BasicDBObject newDocument=new BasicDBObject();
    newDocument.append("$set", new BasicDBObject().append("tenlop", "Lớp DH Kỹ Thuật PHẦN
MỀM 10A CLC__"));
    //lọc đối tượng cần cập nhật
    BasicDBObject filter=new BasicDBObject().append("malop", "DHTH10A");

    //cập nhật
    col.updateOne(filter, newDocument);
    System.out.println("OK");
}

```

DELETE

```

void delete(MongoCollection<Document> col) {
    BasicDBObject filter=new BasicDBObject().append("malop", "1a");
    Document doc=col.findOneAndDelete(filter);
    System.out.println(doc);
}

```

THAO TÁC CRUD VỚI ASYNC DRIVER

CREATE

Chèn một document

JSON document	MongoDB document
<pre> { "name" : "MongoDB", "type" : "database", "count" : 1, "info" : { x : 203, y : 102 } } </pre>	<pre> Document doc = new Document("name", "MongoDB") .append("type", "database") .append("count", 1) .append("info", new Document("x", 203) .append("y", 102)); </pre>
<pre> void insert(MongoCollection<Document> collection) { Document doc = new Document("name", "MongoDB") .append("type", "database") .append("count", 1) .append("info", new Document("x", 203) .append("y", 102)); collection.insertOne(doc, new SingleResultCallback<Void>() { @Override public void onResult(final Void result, final Throwable t) { System.out.println("Inserted!"); } }); } </pre>	

```
}
```

```
collection.insertOne(doc, (Void result, final Throwable t) -> System.out.println("Inse  
Using Lambda Expression
```

Chèn nhiều tài liệu

```
void insertMany(MongoCollection<Document> collection) {  
    List<Document> documents = new ArrayList<Document>();  
    //add your document to documents list  
  
    collection.insertMany(documents, new SingleResultCallback<Void>() {  
        @Override  
        public void onResult(final Void result, final Throwable t) {  
            System.out.println("Documents inserted!");  
        }  
    });  
}
```

READ

Đếm số Documents trong một Collection

```
collection.count(  
    new SingleResultCallback<Long>() {  
        @Override  
        public void onResult(final Long count, final Throwable t) {  
            System.out.println(count);  
            latch.countDown();  
        }  
    });
```

Tìm một tài liệu theo một tiêu chí nào đó

Ví dụ với zip collection, ta tìm các thành phố có dân số > 100000 và hiển thị tài liệu đầu tiên

```
//{"pop":{"gt":100000}}  
collection.find(gt("pop", 100000)).first(  
    (Document doc, Throwable t)->{  
        System.out.println(doc.toJson());  
        latch.countDown();  
    });
```

Tìm một danh sách các tài liệu theo tiêu chí nào đó

```
collection.find(gt("pop", 100000)).forEach(  
    (Document document)->{  
        System.out.println(document.toJson());  
    },  
    (Void result, Throwable t)->{  
        latch.countDown();  
    }  
);
```

Sắp xếp

```
collection.find(gt("pop", 100000)).sort(ascending("pop")).forEach(  
    (Document document)->{  
        System.out.println(document.toJson());  
    },  
    (Void result, Throwable t)->{  
        latch.countDown();  
    }  
);
```

UPDATE

Field Update Operators

Name	Description
<u>\$inc</u>	Increments the value of the field by the specified amount.
<u>\$mul</u>	Multiplies the value of the field by the specified amount.
<u>\$rename</u>	Renames a field.
<u>\$setOnInsert</u>	Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
<u>\$set</u>	Sets the value of a field in a document.
<u>\$unset</u>	Removes the specified field from a document.
<u>\$min</u>	Only updates the field if the specified value is greater than the existing field value.
<u>\$max</u>	Only updates the field if the specified value is less than the existing field value.
<u>\$currentDate</u>	Sets the value of a field to current date, either as a Date or a Timestamp.

```
collection.updateOne(
    eq("i", 10), //condition
    new Document("$set", new Document("i", 110)), //update value
    new SingleResultCallback<UpdateResult>() {
        @Override
        public void onResult(final UpdateResult result, final Throwable t) {
            System.out.println(result.getModifiedCount());
            latch.countDown();
        }
    });
```

```
collection.updateMany(
    lt("i", 100),
    new Document("$inc", new Document("i", 100)),
    new SingleResultCallback<UpdateResult>() {
        @Override
        public void onResult(final UpdateResult result, final Throwable t) {
            System.out.println(result.getModifiedCount());
            latch.countDown();
        }
    });
```

DELETE

```
collection.deleteOne(
    eq("i", 110), //condition
    new SingleResultCallback<DeleteResult>() {
        @Override
        public void onResult(final DeleteResult result, final Throwable t) {
            System.out.println(result.getDeletedCount());
            latch.countDown();
        }
    });
```

```
collection.deleteMany(
    gte("i", 100),
    new SingleResultCallback<DeleteResult>() {
        @Override
        public void onResult(final DeleteResult result, final Throwable t) {
            System.out.println(result.getDeletedCount());
            latch.countDown();
        }
    });
```



```
});
```

BÀI TẬP

BÀI 1

Thực hiện lại các đoạn code phân hướng dẫn

BÀI 2

Cho tài liệu JSON có cấu trúc như hình sau

The screenshot shows a JSON Viewer interface with a tree view on the left and a code editor on the right. The tree view shows the following structure:

- JSON
 - Object
 - firstName : John
 - lastName : Smith
 - age : 25
 - address
 - Object
 - streetAddress : 21 2nd Street
 - city : New York
 - state : NY
 - postalCode : 10021
 - phoneNumbers
 - Array
 - Object
 - type : home
 - number : 212 555-1234
 - Object
 - type : fax
 - number : 646 555-4567

The code editor shows the following JSON code:

```
1 {  
2   "firstName": "John",  
3   "lastName": "Smith",  
4   "age": 25,  
5   "address": {  
6     "streetAddress": "21 2nd Street",  
7     "city": "New York",  
8     "state": "NY",  
9     "postalCode": 10021  
10  },  
11  "phoneNumbers": [  
12    {  
13      "type": "home",  
14      "number": "212 555-1234"  
15    },  
16    {  
17      "type": "fax",  
18      "number": "646 555-4567"  
19    }  
20  ]  
21 }
```

Hãy viết một lớp với các phương thức thực hiện các hành vi Create, Update, Delete

Các phương thức Read với các yêu cầu sau:

- Tìm những person đang sống tại 1 bang cho trước
- Tìm số điện thoại của 1 person cho trước
- Tìm những person có tuổi trên 50
- Tính độ tuổi trung bình cho mỗi bang
- Tính tổng số person cho mỗi bang

BÀI 3

****** Viết chương trình cho phép import dữ liệu từ một bảng cơ sở dữ liệu quan hệ chọn trước vào MongoDB.